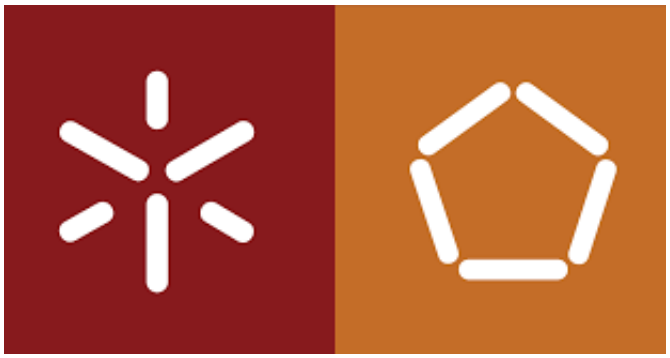


PROBUM

Requisitos e Arquiteturas de Software-3ª Fase

Grupo 8-B

2023/2024

14 de janeiro de 2024

Elementos do Grupo e Contribuição

Número de Aluno	Nome
PG53951	João Pedro Vilas Boas Braga
PG51636	Nsimba Teresa Armando
PG53791	Duarte Gonçalves Parente
PG50006	Juliana Ngoma Cesar Silvério
PG52672	André Lucena Ribas Ferreira
A94942	Miguel Velho Raposo
A91775	José Pedro Batista Fonte
PG52675	Carlos Eduardo da Silva Machado
PG52693	Maria Francisca Mendes Lemos
A96075	João Bernardo Teixeira Escudeiro
PG54162	Rafael Picão Ferreira Correia
PG54093	Miguel Ângelo Silva Senra


PG53951


PG51636

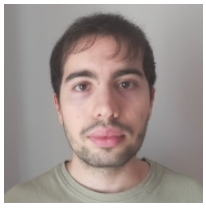

PG53791


PG50006


PG52672


A94942


A91775


PG52675


PG52693


A96075


PG54162


PG54093

Aluno	Cont.	Aluno	Cont.	Aluno	Cont.
PG53951	+0.4	PG52672	+0.4	PG52693	+0.4
PG51636	-2	A94942	+0.4	A96075	+0.4
PG53791	+0.4	A91775	+0.4	PG54162	+0.4
PG50006	-2	PG52675	+0.4	PG54093	+0.4

Tabela 1: Contribuição por Aluno

Prefácio

É com satisfação que apresentamos a terceira fase do projeto PROBUM, um esforço colaborativo e dedicado que visa otimizar processos e aprimorar a experiência tanto para alunos quanto para docentes no contexto acadêmico. Ao longo desta jornada, buscamos viabilizar a realização de avaliações acadêmicas, superando desafios associados a infraestruturas limitadas em instituições de ensino superior.

Nesta terceira fase, nosso foco principal foi implementar um Produto Mínimo Viável (MVP), proporcionando funcionalidades essenciais que atendem às necessidades fundamentais do sistema. O MVP destina-se a permitir que docentes criem provas, alunos as realizem, correções automáticas sejam acionadas e que os alunos possam consultar suas respostas e avaliações.

Ao longo deste processo, utilizamos uma abordagem baseada em micro-serviços, explorando tecnologias familiares e eficientes para cada componente. Adotamos o padrão Observer na notificação de alunos sobre alterações nas provas, uma escolha que reflete nosso compromisso com boas práticas de design.

Índice

Introdução e Objetivos

O Probum visa primariamente a otimização de processos e a melhoria da experiência tanto para alunos quanto para docentes. Em linhas gerais, a iniciativa propõe viabilizar a realização de avaliações acadêmicas por estudantes de uma determinada unidade curricular em instituições de ensino superior, sejam universidades ou politécnicos. Isso é possível mesmo quando as infraestruturas informáticas dessas instituições apresentam limitações em termos de dimensão, disponibilidade e capacidade.

Nesta fase, é esperado no mínimo um conjunto de funcionalidades base, um Produto Mínimo Viável (MVP). Desse modo, de acordo com as fases anteriores é esperado que no mínimo o sistema permita:

- **Criação de Provas por Docentes:** Os docentes devem ser capazes de criar provas compostas por questões de escolha múltipla.
- **Realização de Provas por Alunos:** Os alunos devem poder responder a uma prova, ou seja, realizar uma avaliação.
- **Correção Automática por Docentes:** Os docentes têm a capacidade de acionar a correção automática das respostas fornecidas pelos alunos.
- **Consulta de Respostas e Avaliação por Alunos:** Os alunos podem consultar as respostas dadas a uma prova e sua respetiva avaliação.
- **Notificação aos Alunos:** Deve existir um sistema de notificação para informar os alunos quando a prova estiver disponível.

  Essas funcionalidades constituem a base do sistema e visam atender às necessidades essenciais dos docentes e alunos, proporcionando uma plataforma funcional e eficiente para a realização de avaliações académicas.

Arquitetura Utilizada

Como definido na fase anterior, o grupo optou por uma arquitetura baseada em microsserviços, na qual foram definidos os seguintes microsserviços e os seus respetivos grupos de desenvolvimento:

- **FrontEnd** - Francisca Lemos *(PG52693)* & José Fonte *(A91775)*
- **Microserviço Gestão de Users** - Carlos Silva *(PG52675)* & André Lucena *(PG52672)*
- **Microserviço Gestão Logística** - Nsimba Teresa *(PG51636)* & Juliana Silvério *(PG50006)*
- **Microserviço Gestão de Provas [Criação]** - Bernardo Escudeiro *(A96075)* & Duarte Parente *(PG53791)*
- **Microserviço Gestão de Provas [Realização]** - João Braga *(PG53951)* & Miguel Senra *(PG54093)*
- **Microserviço Gestão de Provas [Consulta]** - Miguel Raposo *(A94942)* & Rafael Picão *(PG54162)*

Tecnologias Utilizadas

Dada a natureza da arquitetura é possível utilizar diferentes tecnologias em cada microsserviço, desse modo, cada grupo tomou as seguintes decisões em cada microsserviço.

FrontEnd

O grupo optou por utilizar o *React* para o desenvolvimento do frontend da aplicação devido à sua flexibilidade, eficiência e robustez, assim como, experiência passada com a biblioteca. *React* é uma biblioteca JavaScript mantida pelo Facebook, conhecida por sua abordagem declarativa e componentizada no desenvolvimento de interfaces de usuário. Além disso, a grande comunidade em torno do *React* oferece um amplo suporte, documentação detalhada e uma variedade de recursos que facilitam o processo de desenvolvimento. Dois dos recursos utilizados foram os design systems Ant Design e Material UI de forma a conseguir desenvolver uma interface esteticamente agradável num curto espaço de tempo. No processo de prototipagem e design o grupo utilizou o *Figma* de forma a delinear a UI/UX a ser desenvolvida.

Microserviço Gestão de Users

A nossa implementação é baseada em ferramentas com as quais já possuímos alguma experiência, portanto escolhemos a *framework spring boot* do *Java* e *MySQL* para a base de dados.

Optamos por escolher a *framework spring boot* pois é a *framework standard* para a construção de microsserviços do Java, oferecendo uma estrutura robusta e produtiva para o desenvolvimento de microsserviços, simplificando o processo de desenvolvimento e oferecendo um conjunto de ferramentas "fora da caixa"que permitem integração e agilidade de desenvolvimento.

A escolha de *MySQL* deve-se ao facto de ser uma tecnologia amplamente adotada pelos desenvolvedores e permitir armazenar dados de forma eficiente, oferecendo estrutura para os dados dos utilizadores, tal como seguir a estrutura dada pelos modelos lógicos desenvolvidos pelo grupo.

Micro-serviço Gestão de Logística

Na fase inicial do desenvolvimento, o grupo optou pela tecnologia *NodeJS* como Backend, dada a familiaridade com a linguagem Javascript, quanto à camada de dados foi escolhido o MySQL, dada a experiência passada com a ferramenta.

Micro-serviço Gestão de Provas (Criação)

Após uma conversação inicial, optamos por utilizar ferramentas com as quais já estávamos familiarizados. Assim,na implementação deste micro-serviço, adotamos uma abordagem tecnológica eficiente, optando pelo MongoDB como sistema de gestão e armazenamento de de dados e pelo Node.js para a camada de backend. A escolha do MongoDB foi motivada pela sua natureza NoSQL, que oferece flexibilidade e escalabilidade para lidar com a variedade de dados associados ao sistema. Por sua vez, a utilização do Node.js e do Express proporcionou uma construção ágil e eficaz da lógica de servidor, permitindo-nos desenvolver microsserviços robustos e de fácil manutenção.

Microserviço Gestão de Provas (Realização)

Para o desenvolvimento deste microserviço tentámos utilizar algumas ferramentas com as quais já tínhamos trabalhado e que nos sentíamos mais confortáveis. A escolha recaiu sobre a utilização de Node.Js para a implementação das rotas e respetiva ligação à base de dados. AO contrário do que foi definido na fase 2, optámos por uma base de dados noSQL, o MongoDB.

Microserviço Gestão de Provas (Consulta)

No cerne do nosso microserviço, destaca-se uma abordagem tecnológica engenhosa, cuidadosamente tramada entre MySQL, Node.js e Express. O MySQL desempenha um papel fundamental ao proporcionar uma base robusta para o armazenamento de dados, elevando a confiabilidade e escalabilidade do nosso serviço. Node.js, como plataforma de execução, aprimora a eficiência com

seu modelo assíncrono, garantindo respostas rápidas e uma gestão eficaz de múltiplas conexões simultâneas. Complementando essa sinergia, o Express simplifica o desenvolvimento de rotas e APIs, conferindo agilidade à manutenção contínua do microserviço.

Implementação

FrontEnd

No processo de desenvolvimento da interface o grupo responsável, optou por dividir o processo em duas fases :

- **Prototipagem** : Nesta fase, o objetivo é desenvolver mockups "quase finais", isto é, muito semelhantes ao resultado final esperado, de forma a tomar todas as decisões de UX/UI nesta fase (abordagem BDUP) e depois apenas implementar. Na sua organização ambos os elementos contribuíram para a construção dos mockups, sendo possível ver o seu resultado no seguinte ficheiro **Figma**.
- **Desenvolvimento**: Nesta fase, de forma a agilizar o processo, o grupo dividiu o FrontEnd do Aluno e do Docente pelos elementos *A91775* e *PG52693*, respetivamente.

FrontEnd do Docente: O FrontEnd destinado ao Docente apresenta as funcionalidades essenciais, seguindo a abordagem MVP. Permite ao Docente criar provas compostas por questões de escolha múltipla e espoletar a correção automática de uma prova. Além disso, é possível consultar uma lista completa de todas as provas já criadas. Ao acessar a uma prova específica, o Docente pode visualizar as questões e as respetivas respostas, nessa mesma página o Docente espoleta a correção automática da prova, indicando o ID da mesma. Estas funcionalidades foram desenvolvidas utilizando ReactJS com a ajuda da biblioteca Material UI.
FrontEnd do Aluno: O FrontEnd do Aluno tem como funcionalidades básicas implementadas as pedidas no MVP. Dessa forma é possível consultar a lista de provas que pode responder, responder a uma prova, consultar a lista de provas concluídas e consultar uma prova concluída e já corrigida. O desenvolvimento destas funcionalidades deu-se com o apoio da biblioteca de componentes *Ant Design* que facilita a criação de tabelas, headers, butões e todos os elementos da UI. Os pedidos HTTP são todos feitos à *API gateway*, na qual são centralizadas todas as rotas, que nos casos do pedidos GET retornam um ficheiro *JSON*, que é utilizado para popular os diferentes campos da UI.

Microsserviço Gestão de Users

O Microsserviço foi construído tendo em conta a sua vertente computacional, onde recebe pedidos da API Layer e os processa, e a base de dados onde se registam tanto os utilizadores como as notificações a que têm acesso.

Nesse sentido, de acordo com os *use cases* que foram selecionados para serem cumpridos funcionalmente pelo microsserviço, identificamos os seguintes como sendo os requisitos funcionais que teríamos de satisfazer: *F1, F2, F3, F4, F5, F6, F7, F8, F9, F10, F11, F12, F13, F20, F47, F61, F67 e F68*.

Para a sua implementação, então, tivemos atenção a duas vertentes distintas, depois de analisar os requisitos:

- **Gestão dos dados de Users** - tendo em conta todas as suas particularidades, nomeadamente registo de novos utilizadores, qualquer que seja o seu tipo, criação e alteração das suas palavras-passe, a sua autenticação e modificação dos seus dados pessoais;
- **Gestão de Notificações** - especialmente aquelas referentes a que aos Alunos recebem quando uma Prova sofre alguma alteração. As funcionalidades referem-se a conseguir as notificações de um utilizador, todas as notificações associadas a uma prova, criar novas notificações e apagar notificações.

Por fim, implementamos rotas que seriam necessárias, para que o nosso microsserviço conseguisse realizar todas operações definidas anteriormente em conjunção com os outros microsserviços. Definimos então as seguintes rotas, algumas distintas das definidas na fase anterior:

Método	Rota	Descrição
GET	/api/usr/profile/<IDU>	Perfil de um utilizador do Sistema
GET	/api/usr/notifications/<IDU>	Notificações de um aluno
POST	/api/usr/login/<password>,<IDU>	POST do formulário de Login
POST	/api/usr/register/<TU>	POST do formulário de Registo
POST	/api/usr/profile/	Atualiza as informações de um utilizador
PUT	/api/usr/verify/<INFO>	Encontra todos os utilizadores que verifiquem uma dada informação
PUT	/api/usr/notifications/<EXAM>, <SUBJECT>	Adiciona uma notificação ao exame dado
POST	/api/usr/notifications/<EXAM>	Adiciona um novo exame à lista de notificações
DELETE	/api/usr/notifications/<EXAM>	Remove um exame da lista de notificações

Para conseguir estabelecer uma conexão entre os utilizadores e as mensagens da notificação, tal como conseguir um registo de todas as provas para conseguir montar o *Observer*, explicado posteriormente, alteramos ligeiramente a estrutura lógica da base de dados, para a seguinte:

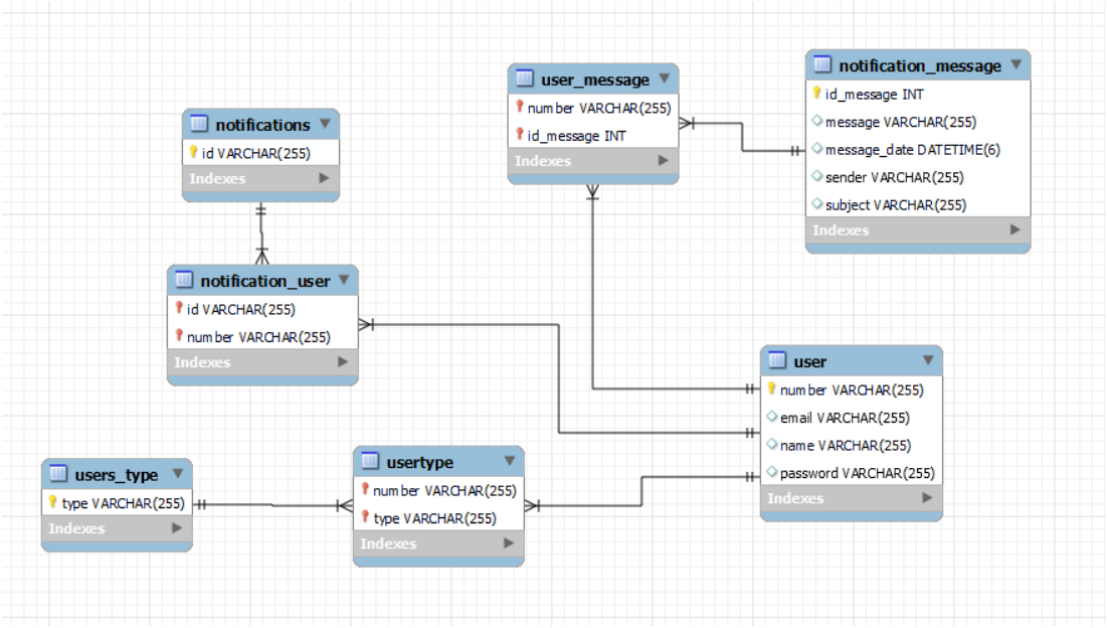


Figura 1: Diagrama Lógico da DB MySQL

Os diagramas das classes Java realizaram-se nas seguintes imagens:

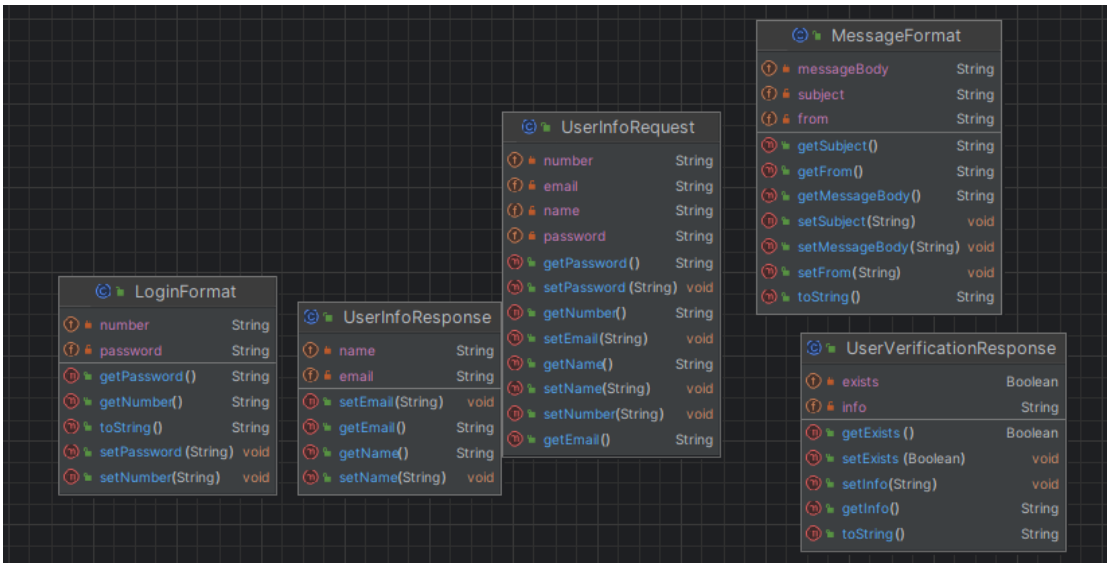


Figura 2: Classes da Autenticação

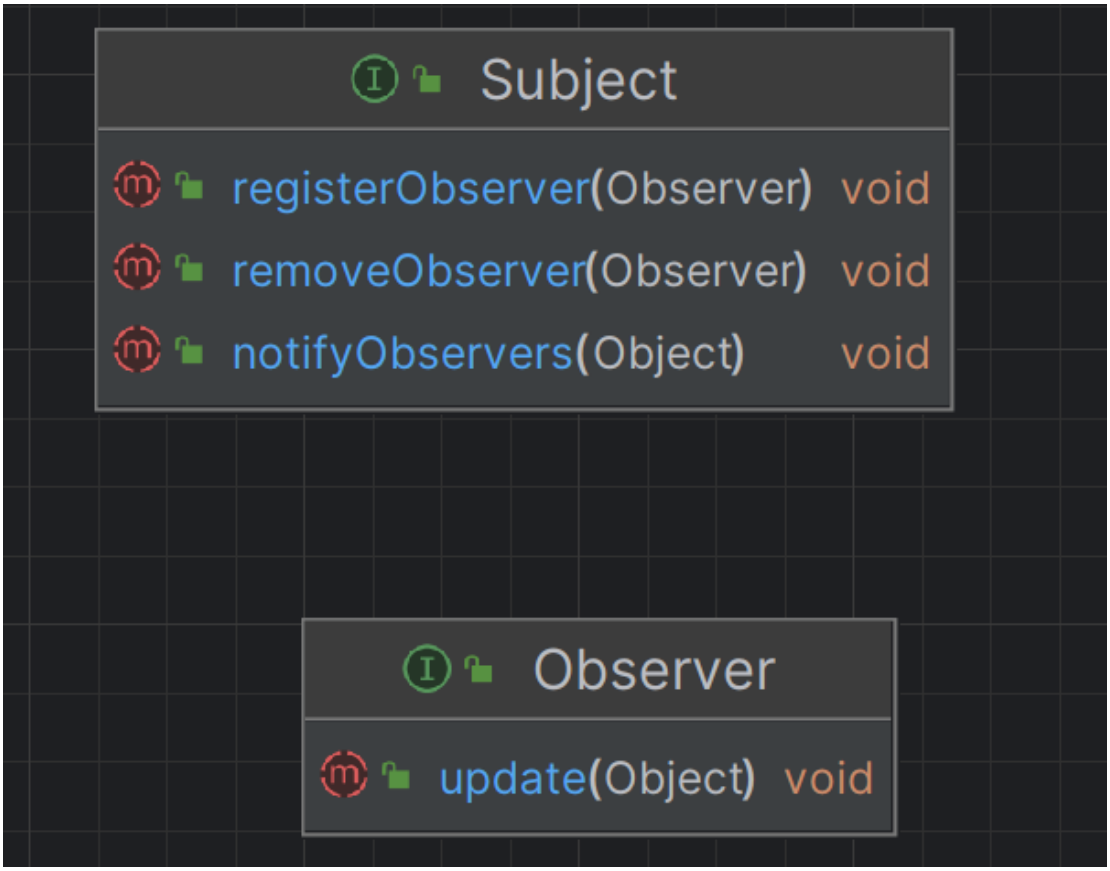


Figura 3: Classes do Padrão Observer

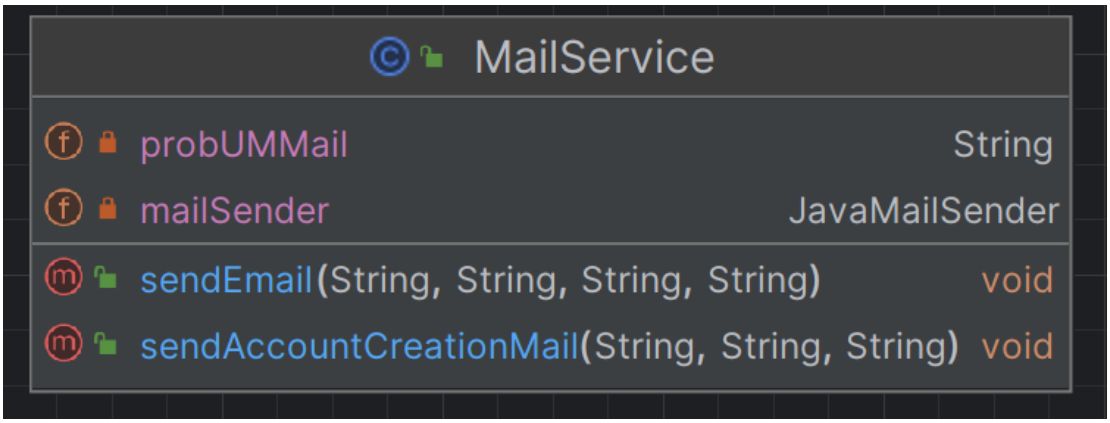


Figura 4: Classe do Serviço de Emails

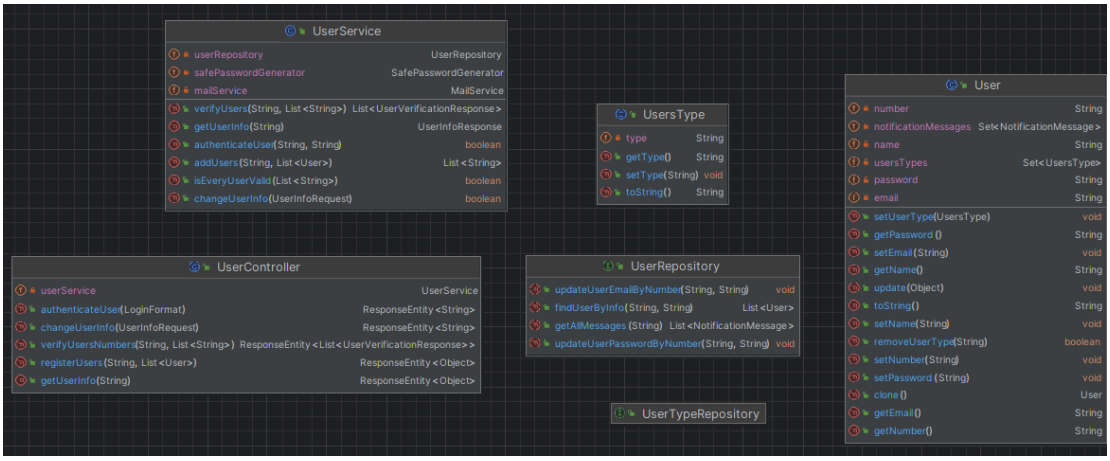


Figura 5: Classes dos Utilizadores

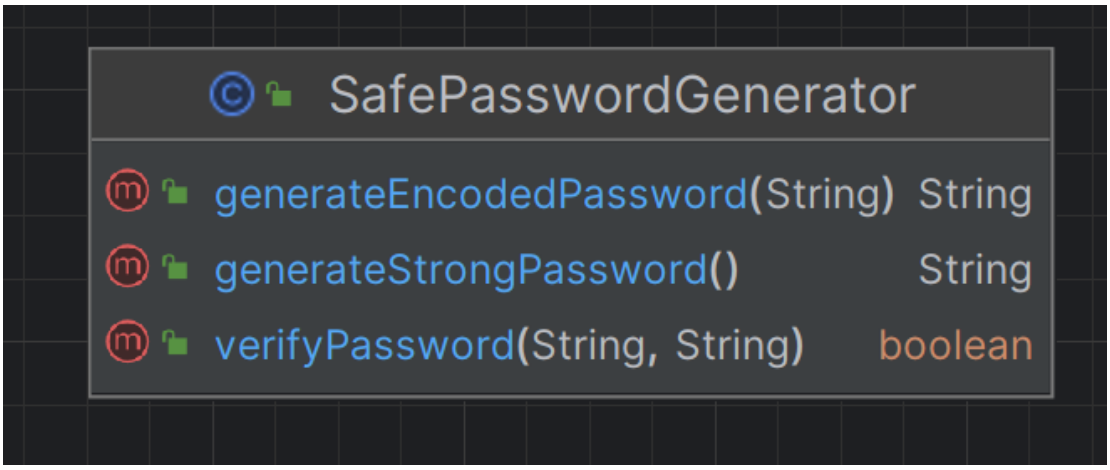


Figura 6: Classe do Gerador de Passwords

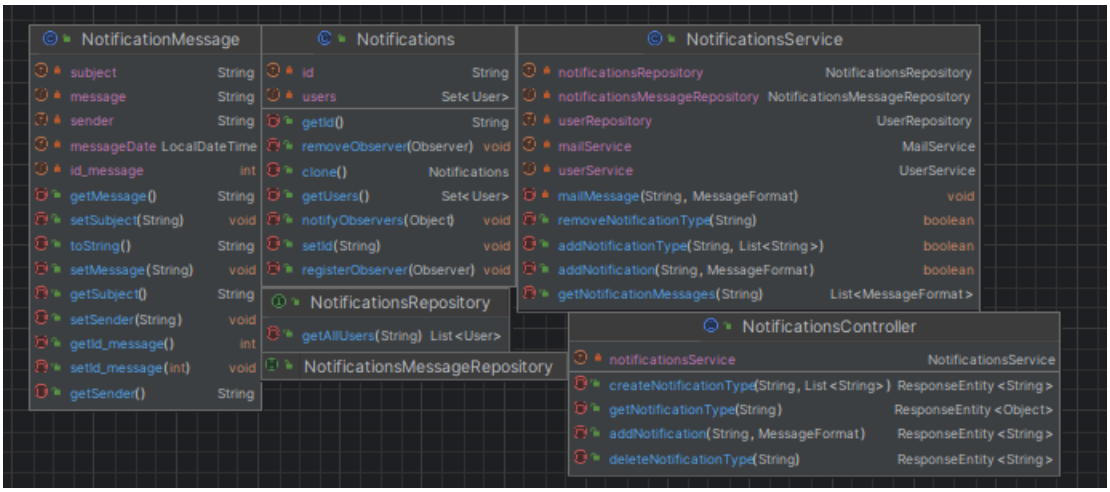


Figura 7: Classes da Gestão de Notificações

Observer

Neste microsserviço, implementamos o padrão de Observer na notificação de alteração dos detalhes de uma Prova. O Sujeito neste caso é o conceito da Prova que foi alterada, onde os Observadores são os Alunos que nela estão registados, e que têm interesse em receber as suas notificações. Uma nova Prova é adicionada à base de dados juntamente à lista dos alunos que lhe pertencem, que ficam registados para ouvirem as alterações que lhe são feitas. Quando uma nova notificação é gerada para essa Prova, uma atualização adiciona essa notificação a todos os alunos que nela estão registada, através da interface *update*.

Isto virtualiza-se no Sujeito **Notifications**, com **User** como Observer.

Microsserviço Gestão de Logística

O grupo responsável pelo microsserviço não conseguiu desenvolver as suas funcionalidades.

Microsserviço Gestão de Provas (Criação)

Baseados na arquitetura da fase anterior, decidimos alterar ligeiramente a arquitetura, tendo em conta a ferramenta escolhida para sistema de armazenamento de dados.Como mongoDB é não relacional, tivemos que alterar ligeiramente as entidades criadas. Na construção da base de dados para a criação da prova construímos o seguinte schema:

Este projeto terá informações relevantes para uma prova como a lista de alunos, os docentes com acesso, o horário, a sala , o curso e a unidade curricular, e para além disso, terá uma lista de outro **Schema** chamado Question schema com as estrutura semelhante à demonstrada na figura abaixo:

Assim, é possível observar que cada questão terá associado um tipo (que no nosso caso será sempre de escolha múltipla) , um enunciado, uma cotação total, bem como uma lista de opções, que como o nome indica, remete para as opções de reposta (alíneas da escolha múltipla).

Cada opção terá um texto associado e uma cotação, que caso seja positiva indica que a resposta é correta, caso seja negativa a resposta é errada.

Dada como concluída a elaboração da base de dados, decidimos passar para a implementação das rotas que seriam necessárias, para que o nosso microserviço conseguisse realizar todas operações definidas anteriormente. Definimos então as seguintes rotas :

```
var criaProvaSchema = new mongoose.Schema({
    _id : String,
    uc : String,
    curso : String,
    cotacao : String,
    nome: String,
    id_docente : String,
    versao : Number,
    data : String,
    hora_preferencial : String,
    tempo_admissao : Number,
    duracao : Number,
    alunos : [String],
    acesso_autorizado : [String],
    questoes : [questaoSchema]
},
```

Figura 8: Schema relativo a criação de uma prova

```
var questaoSchema = new mongoose.Schema({
    _id : String,
    enunciado : String,
    imagem : String,
    cotacaoTotal : String,
    tipo_Questao : String,
    options : [optionsSchema],
})
```

Figura 9: Schema relativo a uma questão

```
var optionsSchema = new mongoose.Schema({
    _id : String,
    texto: String,
    cotacao : String,
    resolucao: String,
})
```

Figura 10: Schema relativo a uma Opção de uma questão

Método	Rota	Descrição
GET	/api/gestao/getprovas/<IDU>	Provas que o docente tem acesso
GET	/api/gestao/gestprovas/<IDU>?id=<IDP>	Obtém uma prova específica
GET	/api/gestao/edit/:<IDP>?idDocente=<ID_DOCENTE>	Obtém detalhes de uma prova para edição
POST	/api/gestao/criar/:<IDU>	Cria uma prova
POST	/api/gestao/criar/questao/<IDU>?idProva=<IDP>	Cria uma questao para uma dada prova
PUT	/api/gestao/gestprovas/edit/:idProva?idDocente=<ID_DOCENTE>	Atualiza uma prova
DELETE	/api/gestao/apagar/:idProva	Elimina uma prova

Terminando a criação das rotas, decidimos criar alguns dados e povoar a base de dados para que fosse possivel verificar o funcionamento das rotas já criadas. Definimos que este microsserviço funcionaria e realiazava as conexões na porta **8010**

Microsserviço Gestão de Provas (Realização)

Tal como aconteceu com outros microsserviços, neste também fizemos a transição de uma base de dados relacional para uma não relacional, neste caso o MongoDB. Esta alteração deveu-se ao facto deste último ter uma implementação mais simples. No entanto, os registos possuem uma estrutura similar àquela apresentada na última fase. Para cada prova realizada guardamos o id da prova, concatenação entre o número de aluno e o id da prova original, o id da prova original, o id do aluno que realizou a prova e as respostas, identificando o id da questão e os ids das respetivas opções. Ficamos com a seguinte estrutura para um registo:

```
var provaDuplicadaSchema = new mongoose.Schema({
    _id: String,
    id_aluno: String,
    id_prova_original: String,
    publico: Boolean,
    respostas: mongoose.Schema.Types.Mixed // map idquestao: {tipo de questao, lista de opçoes/respostas}
})
```

Figura 11: Schema relativo a uma Prova duplicada

Depois de definido o Schema da base de dados, definimos as rotas que registam as alterações nas bases de dados

Método	Rota	Descrição
GET	/api/realizacao/prova	devolve a resolução de uma determinada prova através do id da prova
GET	/api/realizacao/prova_list	devolve a lista de provas da base de dados
GET	//api/realizacao/correct	Envia as respostas a uma dada prova para o serviço de correção
POST	/api/realizacao/save/:idProva	guarda as respostas a uma prova na base de dados
DELETE	/api/gestao/apagar/:idProva	Elimina uma prova

O serviço encontra-se disponível na porta 9999.

Microsserviço Gestão de Provas (Consulta)

Na evolução da arquitetura, optamos por utilizar o MySQL como sistema de gestão de base de dados para o Microsserviço de Consulta de Provas. A estrutura do banco de dados foi mantida em

conformidade com os Modelos Lógicos de Bases de Dados definidos na fase anterior, considerando a natureza relacional do MySQL.

A tabela abaixo descreve as rotas disponíveis, seus métodos correspondentes e suas respectivas funcionalidades.

Método	**Rota**	**Descrição**
GET	/api/correcaoconsulta/correct/listaprova/:id	Retorna a lista de provas corretas de um docente com o ID especificado.
GET	/api/correcaoconsulta/correct/corrAUTOprovas/:id	Retorna as correções automáticas das provas associadas ao docente com o ID especificado.
PUT	/api/correcaoconsulta/correct/corriMANUALprovas/:id	Atualiza as correções manuais das provas associadas ao docente com o ID especificado.
PUT	/api/correcaoconsulta/correct/prova/:id	Atualiza informações de uma prova específica identificada pelo ID.
GET	/api/correcaoconsulta/see/listprovas/:id/ready	Retorna a lista de provas prontas para serem visualizadas pelo docente com o ID especificado.
GET	/api/correcaoconsulta/prova/:id	Retorna informações detalhadas de uma prova específica identificada pelo ID.
GET	/api/correcaoconsulta/alunoready/:id	Retorna informações sobre os alunos que concluíram as provas relacionadas ao docente com o ID especificado.
GET	/api/correcaoconsulta/alunoready/:id/:id2	Retorna informações específicas sobre um aluno que concluiu a prova, identificado pelos IDs.
GET	/api/correcaoconsulta/provas	Retorna a lista de todas as provas no sistema.
POST	/api/correcaoconsulta/provas/	Cria uma nova prova.
PUT	/api/correcaoconsulta/provas/:id	Atualiza informações de uma prova específica identificada pelo ID.
DELETE	/api/correcaoconsulta/provas/:id	Exclui uma prova específica identificada pelo ID.
GET	/api/correcaoconsulta/questoes	Retorna a lista de todas as questões no sistema.
GET	/api/correcaoconsulta/questoes/:id	Retorna informações detalhadas de uma questão específica identificada pelo ID.
PUT	/api/correcaoconsulta/questoes/:id	Atualiza informações de uma questão específica identificada pelo ID.
DELETE	/api/correcaoconsulta/questoes	Exclui todas as questões do sistema.
GET	/api/correcaoconsulta/see/listprovas/:id/ready	Retorna a lista de provas prontas para serem visualizadas pelo docente com o ID especificado.
GET	/api/correcaoconsulta/see/prova/:id	Retorna informações detalhadas de uma prova específica identificada pelo ID.
GET	/api/correcaoconsulta/see/alunoready/:id	Retorna informações sobre os alunos que concluíram as provas relacionadas ao docente com o ID especificado.
GET	/api/correcaoconsulta/see/alunoready/:id/:id2	Retorna informações específicas sobre um aluno que concluiu a prova, identificado pelos IDs.
POST	/api/correcaoconsulta/see/prova/:id/recorrect	Inicia o processo de recorreção de uma prova específica identificada pelo ID.
GET	/api/correcaoconsulta/tipoquestoes	Retorna a lista de tipos de questões disponíveis no sistema.
POST	/api/correcaoconsulta/tipoquestoes	Adiciona um novo tipo de questão ao sistema.
DELETE	/api/correcaoconsulta/tipoquestoes/:id	Exclui um tipo de questão específico identificado pelo ID.

Com essas rotas e a estrutura da base de dados, o Microsserviço de Correção/Consulta de Provas estará preparado para atender às necessidades específicas do seu projeto, utilizando MySQL como sistema de gestão de base de dados. Destaca-se que o microsserviço estará acessível através

da porta 7778.

Desenvolvimento de Funcionalidades

Criação da API

O grupo definiu na fase anterior que existiria uma API capaz de realizar uma conexão entre os microsserviços e o front-end.Assim, construimos uma API com o intuito de apenas realizar a ponte entre os pedidos do frontend e cada um dos microserviços. Esta API servirá como elo de ligação entre todos os microserviços, e garantirá que os pedidos provenientes do Frontend obtêm uma resposta do Backend.

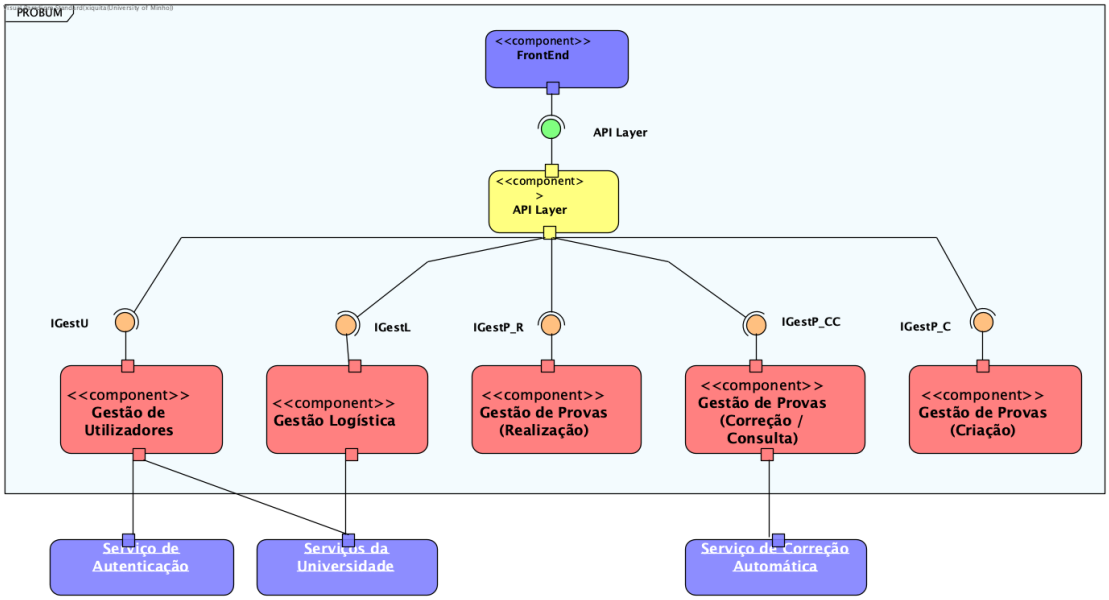


Figura 12: Diagrama de componentes

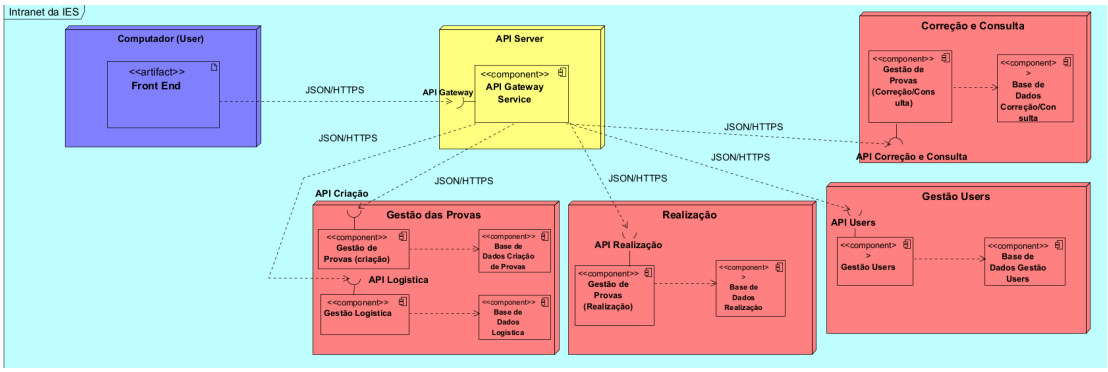


Figura 13: Diagrama de Deployment

Cobertura de Requisitos

Requisito	Prioridade	Feito
F1	Must	✓
F2	Could	
F3	Must	✓
F4	Must	
F5	Must	✓
F6	Must	✓
F7	Must	✓
F8	Must	✓
F9	Must	✓
F10	Must	✓
F11	Must	✓
F12	Should	✓
F13	Could	✓
F14	Must	✓
F15	Must	
F16	Must	

Requisito	Prioridade	Feito
F17	Must	
F18	Must	
F19	Must	✓
F20	Should	✓
F21	Must	
F22	Must	✓
F23	Must	
F24	Must	
F25	Must	
F26	Must	✓
F27	Could	
F28	Should	
F29	Must	✓
F30	Must	✓
F31	Must	
F32	Could	✓

Requisito	Prioridade	Feito
F33	Must	
F34	Must	
F35	Must	
F36	Must	
F37	Must	
F38	Must	
F39	Must	
F40	Must	
F41	Must	
F42	Should	
F43	Should	
F44	Could	
F45	Could	
F46	Could	
F47	could	✓
F48	Must	
F49	Must	✓
F50	Must	

Requisito	Prioridade	Feito
F51	Must	✓
F52	Must	✓
F53	Must	✓
F54	Must	✓
F55	Must	
F56	Must	✓
F57	Must	
F58	Must	✓
F59	Must	
F60	Should	
F61	Should	✓
F62	Must	
F63	Must	
F64	Could	
F65	Must	
F66	Should	
F67	Should	
F68	Should	✓

Requisito	Prioridade	Feito
NF1	Must	✓
NF2	Could	✓
NF3	Must	✓
NF4	Must	✓
NF5	Must	
NF6	Must	
NF7	Must	
NF8	Must	
NF9	Must	✓
NF10	Must	
NF11	Must	
NF12	Should	

Requisito	Prioridade	Feito
NF13	Could	✓
NF14	Must	✓
NF15	Must	
NF16	Must	
NF17	Must	
NF18	Must	
NF19	Must	
NF20	Must	✓
NF21	Must	
NF22	Must	✓
NF23	Must	
NF24	Must	✓

Conclusão

Após o término da terceira fase do projeto de gestão de provas, é evidente que o grupo enfrentou desafios significativos na implementação de todas as funcionalidades planeadas. No entanto, ao avaliar o trabalho desenvolvido não apenas nesta última fase, mas ao longo de todo o projeto, o grupo mantém uma perspectiva positiva em relação aos resultados alcançados.

Internamente, nesta fase específica, reconhecemos que nem todas as funcionalidades desejadas foram completamente implementadas e testadas. Apesar dessas limitações, acreditamos firmemente na capacidade de realizar operações essenciai no sistema. Isso inclui a autenticação, a criação de provas por parte dos docentes, com foco atualmente em questões de escolha múltipla, a realização das provas pelos alunos e a subsequente correção.